

**Historic, Archive Document**

Do not assume content reflects current scientific knowledge, policies, or practices.



aTD223  
.E4  
Copy 2

STATISTICAL ANALYSIS USING STORET AND SAS AT THE  
U.S. EPA NATIONAL COMPUTER CENTER:  
1983 VERSION

BY  
CYNTHIA A. EICHIN

WSDG REPORT  
WSDG-AD-00001  
AUGUST 1983

USDA FOREST SERVICE  
WATERSHED SYSTEMS DEVELOPMENT GROUP  
3825 EAST MULBERRY STREET  
FORT COLLINS, COLORADO 80524

AD-38 Bookplate  
(1-68)

**NATIONAL**

**A  
G  
R  
I  
C  
U  
L  
T  
U  
R  
A  
L**



**LIBRARY**

Document Delivery Services Branch  
USDA, National Agricultural Library  
NAL Bldg  
10301 Baltimore Blvd  
Beltsville, MD 20705-2351

97D223

.F4

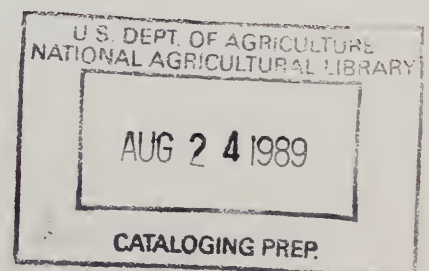
copy 2

## Preface

Since the publication of WSDG Report WSDG-AD-00001 "Statistical Analysis Using SAS at the USEPA National Computer Center", several changes have been made to the computer systems described by that document. Although the basic concepts of STORET data retrieval and statistical analysis using SAS remain the same, the procedures used to accomplish these tasks have changed enough to make an updated document beneficial.

The purpose of this document is to briefly describe the procedures following the system changes. This paper is intended to supersede WSDG-AD-00001, although that paper still offers the forest hydrologist valuable insight regarding the use of statistical procedures to solve water quality problems.

Mention of a company or product by name does not imply endorsement by the U.S. Department of Agriculture.





Page 10

	<u>Page</u>
1.0 Introduction	1
2.0 Introduction to STORET	2
2.1 General	2
2.2 Help	2
2.3 Retrievals	2
2.4 Example Retrieval	3
2.5 Further Computational Files	4
3.0 Introduction to IBM	5
3.1 General	5
3.2 Data Sets at the NCC-IBM	5
3.3 IBM Job Control Language	6
3.3.1 JOB Statement	6
3.3.2 EXEC Statement	7
3.3.3 DD Statement	7
3.3.4 Output Routing	8
3.4 WYLBUR	9
3.4.1 General	9
3.4.2 WYLBUR Signon	9
3.4.3 WYLBUR Commands	9
3.4.4 Help	15
3.4.5 Complete Retrieval and Analysis Using WYLBUR	15
3.5 TSO	16
3.5.1 General	16
3.5.2 TSO Signon	16

3.5.3 TSO and STORET Commands	16
3.5.3.1 STORET Commands	20
3.5.3.2 QED Editor	20
3.5.3.3 TSO Job Submission	24
3.5.4 Help	24
3.5.5 Analysis of SAS Data Set Using TSO	24
3.6 WYLBUR Versus TSO	24
4.0 Introduction to SAS	26
4.1 General	26
4.2 Help	26
4.3 The DATA Step	26
4.3.1 DATA Statement	29
4.3.2 INPUT Statement	30
4.3.3 CARDS Statement	31
4.3.4 INFILE Statement	31
4.3.5 SET Statement	31
4.3.6 MERGE Statement	32
4.3.7 UPDATE Statement	32
4.4 The PROC Step	33
4.5 SAS Analysis via STORET Keywords	33
4.5.1 Introduction	33
4.5.2 Interface Example	34
5.0 References	37
Appendix A: STORET Retrieval and SAS Analysis Using WYLBUR	38
Appendix B: SAS Analysis of Previously Created SAS Data Set Using TSO	55

## List of Boxes

	<u>Page</u>
Box 1. WYLBUR Logon Procedure	10
Box 2. Common WYLBUR Commands	11
Box 3. TSO Signon Procedure	17
Box 4. STORET Retrieval via STORET Commands	18
Box 5. File Creation via QED Editor in TSO	19
Box 6. Common TSO and QED Commands	21
Box 7. WSDG Example Runstreams	27
Box 8. SAS/STORET Interface Example	35



## 1.0 Introduction

The purpose of this paper is to provide Forest Service hydrologists with a basic introduction to the computer systems required to retrieve data from the STORET data base and perform a statistical analysis of that data using the Statistical Analysis System (SAS). After you have finished this document, studied the examples, and referred to the support documents mentioned throughout, you will know how to access the NCC-IBM system via either TSO or WYLBUR and to build jobstreams<sup>1</sup> that will retrieve STORET data and perform a statistical analysis.

This document is not intended to assist the user in the design of his analysis or the interpretation of the output. For a concise review of basic statistics see WSDG-TP-00001 "Statistical Methods Commonly Used in Water Quality Data Analysis."

The three main sections of this document describe the separate parts of the system used to accomplish a statistical analysis of STORET data. The first section describes the STORET data base and the procedures which allow the user to retrieve water quality data. The second section tells about the NCC-IBM, its conventions, and two systems (TSO and WYLBUR) that allow interactive users to access it. The third section describes SAS, a system of programs that allows users to perform statistical analysis using SAS statements.

In order to gain the most benefit from this document, you should have access to the most current STORET USER Handbook, NCC-IBM User's Guide, SAS User's Guide: Basics, and SAS User's Guide: Statistics. These publications will not only provide valuable in-depth insight into the concepts presented in this paper but will also provide informational updates about major system modifications. For example, in mid-November, 1983, NCC anticipates modifying its current 14 character data set prefix (Section 3). This modification, when implemented, will necessitate the following changes throughout this paper:

- (1) USERID  
     OLD - EPAiii  
     NEW - iii
- (2) PREFIX  
     OLD - CN.EPAiii.aaaa (14 character prefix)  
     NEW - iiiaaaa
- (3) SAMPLE JOBCARD  
     OLD - //EPAiii JOB (aaaauuuuu,iiii),'NAME',PASSWORD=(OLD,NEW)  
     NEW - //iii JOB (aaaauuuuu,iiii),'NAME',PASSWORD=(OLD,NEW)

Questions regarding the NCC-IBM system or STORET may be directed to NCC User Services or STORET User Assistance at 1-800-424-9067.

---

<sup>1</sup>A jobstream is a list of JCL (Job Control Language) statements, STORET instructions, SAS statements, input data, or any combination of the above submitted as a unit for processing.

## 2.0 Introduction to STORET

### 2.1 General

STORET is a computerized data base system maintained by the Environmental Protection Agency for the storage and retrieval of water quality data. It consists of the data that have been stored as well as a series of related programs designed to input, retrieve, and analyze the STORET data. The original programs were developed for analysis of water quality data (Water Quality File). However, subsequent programs were developed for other types of data (Municipal Waste Inventory File, Fish Kill File, Contract Awards File, etc.).

The Water Quality File (WQF), the largest STORET data file, consists of (1) station data, which describe the geographical location of a sampling site, and (2) parametric data, which describe sampling conditions and the results of the sample analysis.

Every sampling station is located within a monitoring network that is identified by a unique agency code. Each station is also assigned a primary station code and as many as three secondary station codes. Results of sample analysis are associated with five-digit parameter codes. These codes define the analysis method as well as the form and units of measurement in which the analysis results were stored and may be referenced when retrieving STORET data.

### 2.2 Help

The STORET User Assistance group maintains extensive documentation on the NCC-IBM system. To obtain a listing of the various STORET "HELP" data sets, issue the following command under a TSO session (see Section 3.5):

```
LISTC LEV('STORET.HELP')
```

After the index has been listed, use the LIST or LIST OFFLINE command to print the contents of the members (see Section 3.2).

Information regarding STORET commands can be obtained by issuing the command "HELP STORET" under a TSO session. System response will include a list of STORET commands. If you need more specific information on any of the commands, issue the command "HELP [command name]". (Where examples of commands are presented in this report, capital letters indicate material that you will key in exactly as shown. Lower case letters are used for material you will "fill in" according to your specific need.)

### 2.3 Retrievals

A WQF retrieval request consists of one or more instructions which define what information is to be retrieved and in what format the information should be presented to the user. Usually, an instruction is composed of a keyword and a parameter referred to as a value (separated from the keyword by an equal-sign) which assigns a quantity or other value to that keyword, e.g., PGM=RET.

There are certain formatting conventions that must be followed when coding a retrieval. These include:

- (1) An instruction may begin in any column.
- (2) One or more instructions can be entered on each line.
- (3) A comma must follow each instruction, and embedded commas within values are not allowed.
- (4) An instruction may not be continued from one line to the next, i.e., each line of a retrieval must end with a comma.

## 2.4 Example Retrieval

Following these steps will simplify the coding of your STORET retrieval:

### Step 1: Select Retrieval Program

PGM=program name must appear as the first instruction in any retrieval request because it specifies which retrieval program is to be executed. There are 13 programs available to retrieve station and parametric data from the WQF. Section 5.2 of Part WQ-FS of the STORET User Handbook describes these programs. Since this example will lead to the statistical analysis of actual data values, the RET program is selected.

PGM=RET,

### Step 2: State the Purpose for Your Retrieval

The purpose of this retrieval is training and information, and the recipient of the information is a non-EPA Federal employee. (Refer to Appendix 4 of Part WQ-RET of the STORET User Handbook.)

PGM=RET,PURP=104/FED,

### Step 3: Select Appropriate Stations

Station identification keywords are divided into two categories: selectors and restrictors. At least one selector must be specified and one or more restrictors may also be included to further screen the data. For this retrieval we want to select one station by its agency code and station number, so selector keywords A= and S= will be coded. Other possibilities for station identification are described in Section 4 of Part WQ-RET of the STORET User Handbook.

PGM=RET,PURP=104/FED,A=1110NET,S=23004,

### Step 4: Select Desired Parameters

Appendix B of the STORET User Handbook describes existing parameter codes and how to obtain sorted listings of these codes. The parameters of interest for the example retrieval are temperature (degrees Celsius) and dissolved oxygen:

PGM=RET,PURP=104/FED,A=1110NET,S=23004,P=10,P=300,

### Step 5: Select Program-Specific Keywords

Section 6 of Part WQ-RET of the STORET User Handbook (program descriptions) specifies DR as a keyword that may be used with the RET program to select the location of the decimal point on output. This keyword is parameter dependent and applies only to the preceding parameter code. It is included in the example to retrieve whole numbers as values for parameter 300 (dissolved oxygen).

PGM=RET,PURP=104/FED,A=1110NET,S=23004,P=10,P=300,DR=1,

### Step 6: Select Print Control Instructions

These instructions are used to control what information is printed on the retrieval output and where it is placed. For example, use of the keyword HEAD= is a good way to title retrievals.

PGM=RET,PURP=104/FED,A=1110NET,S=23004,P=10,P=300,DR=1,  
HEAD=EXAMPLE WSDG RETRIEVAL,

Please refer to Appendix A of this document for an example of the complete process required to submit a similar retrieval.

## 2.5 Further Computational Files

The above retrieval request would produce a table of raw data values on the printer. If you intended to use these values as input to SAS, they would either have to be reentered as described in Section 4.3, or, more appropriately, have the STORET retrieval written to an NCC-IBM data set which is referred to as a Further Computational File (FCF). This file can then be used as input to SAS.

The MORE= keyword causes the RET program to write its output into an FCF. There are several values which can be assigned to the keyword MORE=, however, an FCF that was created by the MORE=SAS instruction contains only data records and delimiter records (99 in columns 26 and 27) that are of the same format and length. A file in this format presents the data in a manner which is easily described as input to SAS.

The FCF created as a result of the use of the MORE= keyword is a temporary data set that will not exist after the job ends. The on-line example in CN.EPAWOA.WSDG.SAS(DATASET) illustrates one method you could employ to save the STORET data from the FCF into a SAS data set. This method avoids the necessity of performing another STORET retrieval to perform subsequent SAS analyses on that data.

### 3.0 Introduction to IBM

#### 3.1 General

The STORET data base system and the Statistical Analysis System (SAS) reside on an IBM computer at the EPA's National Computer Center (NCC). Thus, in order to use STORET, SAS, or any other procedure available at NCC, users must be familiar with the various NCC-IBM computer commands that are used to control job processing. The following sections provide a brief summary of some essential NCC-IBM commands and systems.

Section 3.2 describes the NCC-IBM data set naming rules. The data set name identifies a unit of information (either data or commands) to the system. Section 3.3 is concerned with Job Control Language (JCL) statements which direct and control the system while processing a job. Sections 3.4 through 3.6 briefly describe WYLBUR and TSO, two timesharing systems which provide a direct interface between the NCC-IBM system and users of low-speed keyboard terminals.

#### 3.2 Data Sets at the NCC-IBM

In order for the NCC-IBM system to locate and identify data sets stored on the system, each data set must have a unique name. The guidelines for naming data sets, commonly referred to as naming conventions, are outlined below:

- (1) Data set names must be strings of less than 45 characters consisting of segments of less than 9 characters each, separated by periods, e.g.,

CN.EPAWOA.WSDG.SAS

- (2) Segments may not begin with a number and may not end with a period. Otherwise, permissible characters are A-Z, 0-9, @, #, and \$.
- (3) Data set names must be prefixed with the characters CN.EPAiii.aaaa, where iii represents the users EPA initials and aaaa represents the account identifier.

The segments CN.EPAiii.aaaa are collectively referred to as the prefix. The remaining segments in the data set name are referred to as the qualified name. The prefix and the qualified name constitute the fully qualified data set name.

There are two kinds of data sets on the NCC-IBM, sequential and partitioned. Sequential data sets are a single unit and are accessed by the data set name. Partitioned data sets are divided into one or more units referred to as members. Each member is accessed, modified, and recatalogued independently of the other members. Members of a partitioned data set are accessed by the data set name with the member name enclosed in parentheses.

```

|-----prefix-----| |---qualified name---|
      CN.EPAWOA.WSDG.NEW.FILE(member)
|-----fully qualified data set name-----|

```

### 3.3 IBM Job Control Language

Statements contained within a jobstream that direct and control the flow of the job are referred to as Job Control Language (JCL). JCL statements are distinguished from other statements in the jobstream by a slash in columns 1 and 2. The major functions of JCL are to (1) define which program is to be executed, (2) define the sequence of execution, and (3) define the data sets that are required by the program. Examples of JCL include the JOB statement, which identifies the user and provides accounting information for a particular jobstream; the EXEC statement, which identifies the program to be executed; and the DD statement, which specifies input/output devices, volumes, and data sets required by a program. Each job submitted must have a JOB statement, at least one EXEC statement, and a DD statement for each data set that is to be used in a job step.

#### 3.3.1 JOB Statement

The JOB statement marks the beginning of a job. It consists of a slash in columns 1 and 2, followed by the job name (at NCC the job name must be your User-ID, i.e., EPAiii), the keyword JOB, operands, and comments. The following rules must be followed when coding job name:

- (1) It must begin in column 3 following 2 slashes.
- (2) It must consist of 1 to 8 alphanumeric characters (including #, @, and \$), and the first character may not be numeric.
- (3) It must be separated from the keyword 'JOB' by at least one space.

//EPAWOB JOB

The operand field consists of positional parameters and keyword parameters. Positional parameters must be coded first in the operand field in a specific order. Two positional parameters are:

- (1) Accounting Information: All parameters relating to accounting must be separated by commas. If more than one parameter is listed, the entire list must be enclosed in parentheses. This field may take the form (account#,box#).

//EPAWOB JOB (WSDGSTORP,MWOB)

- (2) Programmer Name: This optional parameter must be 20 characters or less and must be preceded by a comma. If there are embedded blanks, the programmer name must be enclosed in single quotes.

//EPAWOB JOB (WSDGSTORP,MWOB),EICHIN

Keyword parameters follow the positional parameters and may be coded in any order. A few of the more commonly used keyword parameters are:

- (3) Message Level: Controls the amount of JCL, allocation, and termination messages output. The value (1,0) will print all JCL, but no allocation or termination messages.

//EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,MSGLEVEL=(1,0)

- (4) Priority: A number ranging from a low priority of 1 through a high priority of 4 that determines the job's rank in the input queue.

```
//EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,MSGLEVEL=(1,0),PRTY=3
```

- (5) Time: Refers to the maximum amount of CPU time a job may use before the job is terminated. Its form is TIME=(minutes,seconds).

```
//EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,MSGLEVEL=(1,0),PRTY=3,  
// TIME=(,30)
```

- (6) Notify: The system will inform the TSO terminal identified by the user's initials when the job completes execution.

```
//EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,MSGLEVEL=(1,0),PRTY=3,  
// TIME=(,30),NOTIFY=EPAWOB
```

- (7) Password: You must specify your current password. This is the same password used to log onto WYLBUR and TSO (Box 1 and Section 3.5.2). The format is PASSWORD=(OLD,NEW). Refer to the NCC-IBM User's Guide for a complete discussion of passwords.

```
//EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,MSGLEVEL=(1,0),PRTY=3,  
// TIME=(,30),NOTIFY=EPAWOB,PASSWORD=(PASS)
```

If a JCL statement exceeds 71 columns, the statement may be continued on another card. The first card must end with a complete parameter (i.e., a comma) and the second card must begin with 2 slashes and a space followed by the remaining parameters.

### 3.3.2 EXEC Statement

The EXEC statement marks the beginning of a job step and identifies the program to be executed to the operating system. It consists of a slash in columns 1 and 2, followed by the step name, the keyword EXEC, operands, and comments. As with the JOB card, the operand field consists of both positional and keyword parameters. However, for the applications discussed in this document, the most common form of this card is:

```
//[stepname] EXEC program name
```

The step name is an optional user-supplied parameter which, if used, should appear in columns 3 through 8. This name should be coded following the same rules as the job name on the JOB card.

Example: //STEP1 EXEC SAS

### 3.3.3 DD Statement

The Data Definition statement describes the data set that is to be used by the program identified by the EXEC statement and specifies the input/output device required for use of the data set. It consists of a slash in columns 1 and 2, followed by the DDname (a name to identify this DD statement), the keyword DD, operands, and comments. The DDname must be unique and is also coded like the job name. The operand field specifically defines the data set to be used. One way to identify a data set that has already been cataloged is:

```
//NAMEA DD DSN=CN.EPAWOA.WSDG.SAS,DISP=SHR
```

For a new data set, the form is much more detailed. Commonly used parameters are described below.

```
//NAMEB DD DSN=CN.EPAWOB.WSDG.NEW.FILE,DISP=(NEW,CATLG,DELETE),
// UNIT=DISK,SPACE=(6160,(60,18),RLSE),
// DCB=(RECFM=VBA,LRECL=137,BLKSIZE=6160)
```

Diagram illustrating the structure of the DD statement operands:

- 1: DSN (Data Set Name)
- 2: DISP (Disposition)
- 3: UNIT (Input/Output Device)
- 4: SPACE (Storage Space Allocation)
- 5: DCB (Data Control Block)

The operands are covered in detail in the NCC-IBM User's Guide, but a general description by section number follows:

- (1) Must be unique and "fully qualified" (See Section 3.2).
- (2) States the disposition of the data set. The first subparameter describes the current status of the data set, the second specifies the action to be taken upon normal job termination, the third specifies desired status upon abnormal termination. If the third subparameter is not specified, it defaults to the second subparameter. For example, if the DISP field is specified as DISP=(NEW,CATLG), DISP=(NEW,CATLG,CATLG) is assumed. If the DISP parameter is not given, DISP=(NEW,CATLG) is assumed.
- (3) Specifies input/output device to be used.
- (4) Describes storage space allocation. See the NCC-IBM User's Guide and SAS User's Guide for details.
- (5) The Data Control Block is also described in detail in the NCC-IBM User's Guide. If the DD statement refers to a SAS data set, the DCB need not be coded.

### 3.3.4 Output Routing

Output from jobs submitted through TSO and WYLBUR is automatically routed to local NCC printers. You can reroute the output to a remote printer with the /\*ROUTE control statement. It is placed in the JCL between the JOB statement and the first EXEC statement. The format is:

```
/*ROUTE PRINT RMTnnn
```

where nnn is the remote printer ID.

Example: //EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,PRTY=3,PASSWORD=PASS

```
/*ROUTE PRINT RMT184
```

```
// EXEC SAS
```

### 3.4 WYLBUR

#### 3.4.1 General

WYLBUR is a timesharing system that allows low-speed terminal users access to the NCC-IBM. WYLBUR provides such functions as text editing, job management, document processing, data set utility functions, and a procedure language.

The WYLBUR system assigns each user a temporary work space referred to as the "active file." Generally, all job stream creation, modification, and preliminary output review are done there. Data sets are created by entering lines of text into the active file where they may be modified using the editing commands. The contents of the active file may then be saved into a data set by issuing the SAVE command, or a copy may be submitted as a job for processing. You may also bring a copy of job output into the active file and scan the output before printing.

#### 3.4.2 WYLBUR Signon

Box 1 outlines the steps required to signon to the NCC-IBM WYLBUR timesharing system. Occasionally when you signon to WYLBUR, the system will issue notification of on-line information which has been stored as News Alerts (see line 12, Box 1). To review these alerts, issue the following commands:

```
USE $PUBLIC.NEWS(ALERTnn) CLEAR
```

```
LIST UNN
```

where nn is the alert number. (The CLEAR operand of the USE command will remove any text currently in the active file. The UNN operand of the LIST command suppresses line numbers when listing the contents of the active file.)

If a WYLBUR session ends abnormally (e.g., telephone line disconnects, etc.), the system saves the contents of the active file in a data set named CN.EPAiii.aaaa.ACTIVE, where iii is the user's EPA initials and aaaa, the account identifier. To recover the contents, reconnect to WYLBUR, following the signon procedure. When WYLBUR prompts you with

```
OK TO RECOVER FROM ACTIVE FILE?
```

respond YES. WYLBUR will then recover the active file and return it to your active area.

#### 3.4.3 WYLBUR Commands

To perform a STORET retrieval and/or a SAS analysis, you must create a jobstream and submit it to the system. To help accomplish this task, the following discussion and Box 2 will briefly describe some of the more commonly used WYLBUR commands.

Box 1

WYLBUR LOGON PROCEDURE

After dialing the TYMNET access number (which may be obtained by dialing 1-800-334-0741), wait for the tone, place the phone in the coupler, then follow these steps to access the WYLBUR text editing facility. (Note: User response is underlined.)

```

1      PLEASE TYPE YOUR TERMINAL IDENTIFIER
2      -2401-151-
3      PLEASE LOG IN:IBMEPA1;NCC
4      P 15
5      IBM1 IS ON LINE
6      OBS
7      OBS WYLBUR 6.0 AT EPA TERMINAL TTY626 12:12:32 09/09/82
8      USER ID?EPAWOB
9      PASSWORD (CURR(/NEW))? XXXXXXXXXX
10     ACCTUID? WSDGSTORP
11     BIN/BOX? MWOB
12     9/8/82: NEW LEVEL OPTIMIZER III           -SEE NEWS ALERT1
13     CN.EPAWOB.WSDG.QWIK NOT ON CATLG 1 LEVELS OK
14     CN.EPAWOB.WSDG.ACTIVE NOT ON CATLG 1 LEVELS OK
15     CN.EPAWOB.WSDG.LIB NOT ON CATLG
16     EXEC BREAK
17     COMMAND?

```

<u>LINE</u>	<u>EXPLANATION</u>
1	Terminal identifiers may be obtained by listing member \$PUBLIC.NEWS(VAN020). However, in general, CRT terminals are type A and thermal printers are type E (both 300 baud).
2	These numbers describe how this session is connected to the IBM.
3	Before keying in IBMEPA1;NCC (the host name and password), you should enter control H to suppress echoing and control P to ensure even parity.
4	This number is the host computer port of this connection.
5	Response advises that you have gained access to the NCC-IBM.
6	At this point, decide whether to use TSO or WYLBUR. For this example, WYLBUR is chosen. (NCC-IBM WYLBUR is supported by On-Line Business Systems, Inc. - hence the OBS designation.)
7	Response advises successful access to WYLBUR.
8-11	Computer prompts for user identification, authorization, and accounting information.
12	On-line news broadcast.
13-16	WYLBUR messages indicating current status of user's catalog.
17	You are in WYLBUR in the command mode.

COMMON WYLBUR COMMANDSEDITING COMMANDS

CHANGE - alters text in the active file. This command shifts text to the right or left.

example: CHANGE 'a b' TO 'c d' in 3

LINE 3  
before: 1 2 a b c d 3 4  
after: 1 2 c d c d 3 4

CLEAR - removes all information from the active file.

example: CLEAR TEXT

COLLECT - puts the terminal into input mode. Break key returns the terminal to command mode.

example: COLLECT

COPY - duplicates lines to specific line numbers.

example: COPY 2 TO 5

DELETE - removes lines from the active file.

examples: DELETE 2 (deletes line 2)  
DELETE 7/10 (deletes lines 7 through 10)

HELP - provides user with description of commands.

example: HELP HELP

INSERT - inserts lines into the active file.

example: INSERT 2.1 2.2 (prompts for 2 lines to insert)

LIST - displays lines from active file.

example: LIST UNN (lists entire data set without line numbers)

MOVE - similar to COPY except text is deleted after it is copied.

example: MOVE 2 TO END

NUMBER - rennumbers active file.

example: NUMBER (rennumbers the entire data set)

REPLACE - replaces text on an existing line.

example: replace 7  
or type the line number, a space, and the new text

## DATA SET COMMANDS

SAVE - catalogs a copy of the active file.

examples: SAVE NEWFILE  
          SAVE MYFILE REPLACE   (replaces previously saved  
                                  MYFILE with a copy of the  
                                  active file)

SCRATCH - deletes a data set that has been saved.

example: SCRATCH MYFILE

USE - copies a data set from disk storage into your active file.

example: USE MYFILE

## JOB SUBMISSION AND RETRIEVAL COMMANDS

FETCH - retrieves queued output

example: FETCH job#

LIST OFFLINE - lists contents of active file at destination specified.

example: LIST OFFLINE DEST RM# COPIES n

LOCATE - determines status of batch jobs.

example: LOCATE

PRINT - causes job awaiting output or fetch to be released for printing.

example: PRINT job# DEST RM#

PURGE - deletes jobs from input or output queue.

example: PURGE job#

ROUTE - changes output routing.

example: ROUTE job# DEST RM#

RUN - submits active file to the input queue. The fetch operand causes output to be held.

example: RUN FETCH

You will notice as you read this section that WYLBUR commands requiring data set names as operands (e.g. USE and SAVE) don't necessarily require you to supply a "fully qualified" data set name. If, during processing, you enter a data set name preceded by a dollar sign, WYLBUR assumes the supplied name is fully qualified. If you enter a name without the dollar sign, WYLBUR qualifies it fully by appending the prefix

CN.EPAiii.aaaa

where EPAiii and aaaa are the user id and account number given during the logon process (Box 1). For example, if you logged on with the user id EPAXYZ and account A123, WYLBUR will interpret the command

USE MYFILE

as

USE \$.N.EPAXYZ.A123.MYFILE

As shown in Box 1, Line 17, after gaining access to WYLBUR, the system is in command mode and is ready to receive an instruction. To create a jobstream several WYLBUR text editing commands will have to be used. The COLLECT command, which shifts WYLBUR from command mode to collect (input) mode, is usually the first command issued when creating a new jobstream. You can also switch modes (command to collect or collect to command) by depressing the break or attention key on the terminal.

Once WYLBUR is in collect mode, begin typing the jobstream. A carriage return at the end of a line indicates the end of an input image. Then WYLBUR will prompt for the next input line. If you make a typing error during input, it can be corrected by simultaneously depressing the control and H keys once for each preceding character to be deleted. This procedure allows the user to backspace to the point of error and correct the input. It is important to note that you cannot correct the line if it has already been transmitted by depressing the return key. To delete or correct an entire line, depress the break key.

When entry of the jobstream is complete, hit the break key to return to command mode. Now you may use any of the WYLBUR editing commands to modify the jobstream. Box 2 lists many of the common WYLBUR editing commands.

Once the jobstream is completed, you may want to save it in a data set on the NCC-IBM system. The SAVE command will catalog a copy of the active file in a data set under a user-supplied name.

SAVE filename ON USERnn

where nn is the number of one of the disk packs available for public use (nn may be any number between 03 and 49 inclusive). The data set will be cataloged with the fully qualified data set name

CN.EPAiii.aaaa.filename

To process the jobstream, submit a copy of the active file for processing by issuing the RUN command. It is usually a good idea to review output to make sure the job has run correctly before it is printed. The FETCH operand of the RUN command will prevent the output from being queued for printing and place it in hold status for a maximum of 48 hours. Its form is:

### RUN FETCH

If the RUN command is issued without the FETCH operand, job output will automatically be queued for printing following execution.

After you issue the RUN command, the system will respond with a message indicating: (1) that the job has been submitted for processing, and (2) the identification number that was assigned to the job.

The LOCATE command is used to check the status of your job once it has been submitted. The system's response and its descriptors follow:

COMMAND? LOCATE

```
1 JOB 4253 EPAWOB53 AW EXEC C=D POS=200 PRI=07
2 JOB 3405 EPAWOB05 AW FETCH
3 JOB 6568 EPAWOB68 AW OUT RM184
COMMAND?
```

- 1 indicates job 4253 is still awaiting execution
- 2 indicates job 3405 has run and is in output hold awaiting fetch
- 3 indicates job 6568 is queued for printing at the remote specified

Notice that the LOCATE command refers to jobs by both the job number and the job name, e.g.,

```
      1      2
  [-----] [-----]
JOB 4253 EPAWOB53 AW EXEC C=D POS=200 PRI=07
```

- (1) system generated job number.
- (2) user-supplied job name (the numbers appended to the job name are system generated to insure each job has a unique name).

The FETCH command is used to bring a copy of the output into the active space. Its form is

### FETCH job#

After the output is retrieved, you can determine whether your job was successful by using the LIST editing command to examine the output. Two possible items to look for are error messages and completion codes. For example:

- (1) LIST 'error'

A "VOID RANGE" response indicates that no line containing the character string 'error' exists in the output.

- (2) LIST 'cc'

The NCC-IBM system outputs a completion code (cc) after each step. A zero completion code usually indicates success, however, a completion code other than zero does not always indicate job failure. See the OS/VS Message Library for a more detailed description of these codes.

If the job was successful, use the command

```
PRINT job# DEST RM#
```

to queue the output to the remote printer specified by RM#.

If the job was unsuccessful, it can be deleted with the PURGE command:

```
PURGE job#
```

Now bring a copy of the jobstream back into the active space with the USE command; make corrections and resubmit it. As previously mentioned, the CLEAR operand with the USE command will remove any text that is currently occupying the active space.

```
USE filename CLEAR
```

#### 3.4.4 Help

A good way to learn how to use WYLBUR is to use OBS's on-line training package. To begin a training session, log onto WYLBUR and enter this command:

```
EXECUTE FROM $WYLPUB.LEARN(LEARN)
```

In addition to this training routine, the WYLBUR Reference Manual is stored on-line in the data set PUBLIC.OBSWYL.DOC(REFMAN). Use the LIST OFFLINE command to obtain a copy of this documentation.

#### 3.4.5 Complete Retrieval and Analysis Using WYLBUR

Statistical procedures are often based on the assumption that the data sample comes from a population that is normally distributed. For this reason, it is often necessary to perform a statistical procedure (like PROC UNIVARIATE) to test data for normality. Appendix A includes a listing of a job that performs this test by using WYLBUR to modify an on-line example data set. This jobstream performs a STORET retrieval, builds and catalogs a SAS data set, and does a PROC CONTENTS to document the creation of the data set, then a PROC UNIVARIATE to test for normality. The output produced by submission of the jobstream is also included.

## 3.5 TSO

### 3.5.1 General

TSO, like WYLBUR, is a timesharing system that allows users of low-speed terminals access to the NCC-IBM facilities. Features of TSO include: (1) text editing, (2) a facility which allows users to execute programs that require terminal input and return output to the terminal, (3) a command procedure that allows users to invoke a data set containing executable TSO commands as a single command (CLIST procedure), and (4) numerous STORET commands (CLISTs) that are available to TSO users with STORET accounts. The NCC version of TSO has an IBM edit capability as well as a line editor (QED). The QED editor was used to edit the examples in the following sections.

### 3.5.2 TSO Signon

TSO's signon procedures are the same as for WYLBUR's until you specify which system to access (see Box 3). At that point, enter TSO instead of OBS. To the LOGON prompt, key in

```
LOGON userid/password ACCT(account number)
```

The system will respond with several informative messages, and finally the word READY. This prompt indicates that the logon procedure is complete and TSO is ready to receive the first command. As in WYLBUR (Section 3.4.2), when you logon to TSO, you will be notified of on-line system messages. To list a message in TSO, issue this command:

```
NEWS ALERTnn
```

where nn is the News Alert number.

If a TSO session terminates abnormally (e.g., the telephone is disconnected), follow this procedure to reconnect: (1) redial and follow the logon procedure up to the LOGON prompt, (2) then type

```
LOGON userid/password RECON
```

The reconnected session picks up where the previous session was terminated, e.g., if it was in edit mode, the new session will be returned to the editor and any changes made before the disconnection will be intact.

### 3.5.3 TSO and STORET COMMANDS

TSO provides two ways to create jobstreams to retrieve STORET data. You can use either STORET commands or the QED edit commands to create a jobstream.

Box 3

TSO SIGNON PROCEDURE

-2401-153-

PLEASE LOG IN:IBMEPA1;NCC

P 16

IBM1 IS ON LINE

TSOENTER LOGONLOGON EPAWOB/PASS ACCT(WSDGSTORP)ICH7000II EPAWOB LAST ACCESS AT 12:50:28 ON TUESDAY, SEPTEMBER 14, 1982

EPAWOB LOGON IN PROGRESS AT 12:53:53 ON SEPTEMBER 14, 1982

NO BROADCAST MESSAGES

WELCOME TO TSO

YOU ARE NOW IN THE STORET ENVIRONMENT

READY

(You are now in TSO and may use any TSO feature, see Boxes 4 and 5.)

Box 4

STORET RETRIEVAL VIA STORET COMMANDS

(Note: User response is underlined.)

READY  
%RET

WELCOME TO STORET  
 ENTER YOUR RETRIEVAL REQUEST AFTER THE PROMPTING LINE NUMBERS.  
 TO TERMINATE INPUT, ENTER A -NULL LINE- (CARRIAGE RETURN)  
 IN RESPONSE TO LINE NUMBER PROMPT

ENTER A NAME FOR THE WORKING STORAGE - FILES

INPUT  
 00010 PGM=RET,PURP=104/FED,A=1110NET,S=070006,P=10,  
 00020

USE ANY EDITING COMMANDS DESIRED.  
 USE THE '%SCAN' COMMAND TO SUBMIT THIS RETRIEVAL  
 AFTER YOU HAVE 'SAVED' IT.

WORKING STORAGE IS NAMED  
 FILES

QED  
END SAVE  
SAVED  
 READY  
%SCAN FILES  
 SCAN BEGINS .... DATA SET NAME "FILES"

00000010 PGM=RET,PURP=104/FED,A=1110NET,S=070006,P=10,  
 00000011 ./EPAWOB JOB (WSDGSTORPUUU,MWOB),STORET,NOTIFY=EPAWOB,TIME=5,  
 00000012 ./ MSGLEVEL=(1,1),PRTY=1  
 00000013 \*\*ROUTE PRINT HOLD  
 00000014 \*\*JOBPARM LINES=10

NO SYNTAX ERRORS FOUND IN RETRIEVAL

JOB 2072 SUBMITTED  
 ... SCAN ENDS

READY  
QUEUE  
 Q0800 WELCOME TO QUEUE - ENTER: "H" FOR COMMAND LIST

LOC

JOB*	JOBNAME	QUEUE	POSITION	LINES	DESTINATION
2043	EPAWOB	TSO USER	40		
2072	EPAWOB72	INPUT	D, 85		

QUEUE  
 END  
READY  
LOGOFF

Box 5

FILE CREATION VIA QED EDITOR IN TSO

(Note: User response is underlined.)

READY

EDIT FILENAME NEW

INPUT

00010 THIS IS A NEW00020 FILE BEING CREATED00030 USING THE QED EDITOR

00040

QED

END SAVE

SAVED

READY

LOGOFF

As in WYLBUR, TSO, QED, and STORET commands requiring data set names as operands don't necessarily require you to supply a "fully qualified" data set name. To specify a fully qualified data set name, enclose the name in single quotes. If the name is not enclosed in single quotes, it will be qualified by appending the prefix

CN.EPAiii.aaaa

where EPAiii and aaaa are the user id and account number given during the logon process. For example, if you logged on with the user id EPAXYZ and account A123, the command

EDIT SAS(DATASET) OLD

is interpreted as

EDIT 'CN.EPAXYZ.A123.SAS(DATASET)' OLD.

### 3.5.3.1 STORET Commands

The STORET command %RET may be used to create the jobstream to perform the retrieval. The command %SCAN may then be used to check the jobstream for errors and to submit it for processing (Box 4). The JCL used by the %RET command (Box 4, Lines 00000011-00000014) resides in the member \$\$JOBPRM of the data set CLIST. To review these values, issue the %JOBPRM command after gaining access to the TSO system. You may then use edit commands to change the desired default values.

### 3.5.3.2 QED Editor

The QED editor has two modes: input and edit. As shown in Box 5, the TSO EDIT command invokes the QED editor. The data set name is included with this command as well as an indication of whether the data set is new or old. If the command specified that the data set was old, the QED editor will be in edit mode; if it indicated the data set was new, it will be in input mode.

The END command terminates the EDIT command. At this point you can decide whether or not to save the edited data set, and enter

END SAVE  
or  
END NOSAVE

A carriage return on a blank line (null line) switches the terminal from one mode to the other. As in WYLBUR, errors may be corrected by using control and H keys for backspacing or the break key to eliminate the current line. Box 6 presents a brief list of common TSO and QED commands.

COMMON TSO AND QED COMMANDSTSO COMMANDS

DELETE: deletes data sets or members of partitioned data sets.

example: DELETE CN.EPAWOA.WSDG.OLDFILE

EDIT: creates or modifies a data set

example: EDIT filename OLD      (modify existing data set)  
          EDIT filename NEW      (create new data set)

HELP: gives full or partial information on commands

example: HELP commandname

LISTDS: lists data set attributes.

example: LISTDS FILE.NAME M (will print volume, members, and storage information)

LISTCAT: lists data sets that have been entered on your catalog

example: LISTCAT

SUBMIT: submits batch job for processing.

example: SUBMIT filename

QUEUE: displays information regarding status of jobs in input and output queues.

EDIT SUBCOMMANDS

CHANGE: modifies characters.

example: CHANGE 15 /wrong/right/ (replaces character strings in line 15)

COPY: duplicates existing lines to another location in the data set.

example: COPY 10 100 (duplicates line 10 before the line following line 100)

DELETE: removes existing lines from data set.

example: DELETE 50 (removes line 50)

END: terminates EDIT command (may use SAVE or NOSAVE option).

example: END NOSAVE

FIELD: restricts scope of subsequent CHANGE, FIND, and LIST commands.

example: FIELD 1 25 (restricts action to columns 1 through 25)

FIND: locates a character string.

example: FIND TON (prints the first line encountered that contains the character string "TON".)

INPUT: adds or replaces lines in the data set.

example: INPUT line-number R (replace existing lines or insert vacant lines)  
INPUT line-number I (insert lines without changing existing lines)

INSERT: adds data lines to the data set.

example: INSERT data (adds "data" after line where pointer is positioned)

(In a data set numbered 10 20 30, "15 data" would add the word data on line 15. "30 change" would replace the contents of line 30 with the word "change".)

LIST: displays lines at terminal.

example: LIST 10 (prints line 10)

MOVE: repositions existing data lines.

example: MOVE 30 10 (changes line sequence from 10 20 30 to 10 30 20)

RENUM: rennumbers the data set.

example: RENUM

SAVE: saves and catalogs the data set.

examples: SAVE  
END SAVE

## QUEUE SUBCOMMANDS

END: exits from QUEUE command.

FETCH: lists output.

example: FETCH jobid

FIND: finds next occurrence of string.

example: FIND 'string'

HELP: explains command or error message.

LOCATE: displays job status for your ID.

PURGE: cancels job from input or execution and purges output.

ROUTE: routes job to specified remote site.

example: ROUTE job# remote#

### 3.5.3.3 TSO Job Submission

Once the jobstream has been created, use the SUBMIT command to schedule the job for processing. Jobstreams may be submitted from the QED editor or from TSO. The command form for QED is SUBMIT, and for TSO, SUBMIT data set name.

If you want to review the output before it is printed, include a card like /\*ROUTE PRINT RMT255 in the jobstream. Remote 255 is a dummy remote site where output can be held until you retrieve it with the FETCH subcommand of the QUEUE command. A detailed explanation of the capabilities of the queue facility can be obtained by issuing the command

HELP QUEUE

If the job is successful, reroute the output with the following command

ROUTE job# remote-destination

### 3.5.4 Help

TSO's HELP command provides on-line information that can be listed at the terminal including documentation about TSO commands, EDIT subcommands, STORET commands, and QUEUE subcommands. During a TSO session, enter this command to get started:

HELP HELP

### 3.5.5 Analysis of SAS Data Set Using TSO

Once the desired STORET data has been retrieved and cataloged in a SAS data set, you can use that data set over and over again as input to a SAS procedure. Appendix B shows, step by step, how TSO can be used to build a jobstream to use a SAS data set as input to the CORR procedure.

## 3.6 WYLBUR Versus TSO

As described in Sections 3.4 and 3.5, WYLBUR and TSO have somewhat similar capabilities. Although TSO has more commands (and is generally more expensive to use) than WYLBUR, every function that can be performed in TSO can also be done with WYLBUR by submitting an appropriate jobstream. Your preference for TSO or WYLBUR will depend on whether or not the convenience of the TSO commands with their nearly immediate responses and higher costs, is more desirable than the additional effort and lower costs of WYLBUR. Examples of the features of each system follow.

- (1) The user will find that he can create, submit, and retrieve jobs more cheaply through WYLBUR than he can using the QED editor and QUEUE command in TSO.
- (2) The editing facility of WYLBUR is more user-friendly than TSO's QED editor.

- (3) NCC provides an on-line tutorial program that enables users to learn how to use WYLBUR with ease in a minimal amount of time.
- (4) Restoring, archiving, and deleting a small number of data sets in TSO is more convenient than submitting jobs to perform these functions through WYLBUR, and the costs are not significantly different.
- (5) There is a wide range of STORET commands available in TSO (but not in WYLBUR) that are designed to simplify the implementation of STORET functions.
- (6) SAS may be executed interactively in TSO with immediate (or nearly so) results.

These are but a few examples. You should become somewhat knowledgeable of both systems in order to make appropriate use of them.

## 4.0 Introduction to SAS

### 4.1 General

SAS is a group of computer programs designed to allow the user to check data for errors, perform statistical analyses, manipulate data, and print the results. All SAS jobs consist of statements that are divided into DATA and PROC steps. A SAS job can consist of one step or more.

When you are coding SAS statements, remember:

- (1) User-supplied SAS names must be less than 9 characters long, and the first character must be a letter or an underscore. Special characters and embedded blanks are not allowed.
- (2) SAS statements usually begin with a SAS keyword, which tells SAS which activity is to be performed (e.g., DATA, PROC, COMMENT), and may contain other information that further describes how the activity is to be performed. SAS statements may begin in any column, more than one statement may appear on one line, and statements may be continued from one line to the next so long as words are not split. All SAS statements must end with a semicolon.
- (3) A blank or a special character (i.e., "=" or ";") must separate each word in a SAS statement.
- (4) Comments may be used to document jobs. These statements may begin with the keyword COMMENT or an asterisk and must not contain a semicolon.

### 4.2 Help

The Watershed Systems Development Group supports an on-line example data set to assist inexperienced users with a few basic SAS procedures. Each example includes a jobstream to do a different type of analysis and is stored as a member of the partitioned data set CN.EPAWOA.WSDG.SAS. Box 7 lists each of the members and gives a short description of its function.

### 4.3 The DATA Step

SAS requires data input to any SAS procedure to be in a specific format. This is accomplished by using the DATA step to create a SAS data set that contains your data. The DATA step includes statements SAS needs to create the new SAS data set and programming statements SAS uses to perform data manipulation during this building process.

WSDG EXAMPLE RUNSTREAMS

DATA SET NAME: CN.EPAWOA.WSDG.SAS

<u>MEMBER NAME</u>	<u>Description of Procedure</u>
AOVONE	One-way analysis of variance using card images as input.
AOVTWO	Two-way analysis of variance using card images as input.
CHART	Bar charts and histograms using a SAS data set as input. Demonstrates how to build histograms by percentages or weighted by certain variables or both.
CLUSTER	Cluster analysis.
CONTENTS	Profile of a SAS data set. Generates a printout showing the variable names, locations, and sizes of SAS data sets, along with the statements that created them.
DATA	Example data for UNIVFILE.
DATASET	Creates a SAS data set from STORET data. Demonstrates the process of generating a Further Computational File (FCF) from a STORET retrieval request, building and cataloging a SAS data set from it.
GLMCLCRD	Linear regression using card images as input. Generates 95% confidence limits about each individual predicted value of the dependent variable and demonstrates the use of the ID statement in the SAS GLM (general linear models) procedure.
GLMCLI	Linear regression using a SAS data set as input. Demonstrates how to select certain stations for analysis, and how to use titles, options, and labels on graphs.
GLMLGLG	Linear regression using a SAS data set as input. Demonstrates computational statements (LOG10 in this case), renaming variables, and page titles.
GLMRGPT	Linear regressions and plots using a SAS data set as input. Demonstrates the use of the SAS MERGE statement, graph labeling, page titling, and generates a plot with overlays.
GLMSAS	Linear regression using SAS statements to merge two stations by common dates of observations and to print the combined data.
MEANST	Paired t-test using a SAS data set as input. Demonstrates the process of building a new variable as the difference between two others, then testing that the values of the new variable are significantly different from zero.

<u>MEMBER NAME</u>	<u>Description of Procedure</u>
MULTPLOT	Multiple plots using a SAS data set as input. Shows the use of the MERGE statement, and how to generate overlayed plots.
NESTED	Two-level nested analysis of variance using card images as input.
PLOTTIME	Concentration-time plots using a SAS data set as input. Demonstrates the use of IF..THEN statements in building new variables, renaming variables, labeling plots, and titling pages.
PRINT	Prints the contents of a SAS data set. Shows the use of the SORT statement.
SOURCE	Generates a printout with a listing of each example member in this data set.
STEPWISE	Stepwise multiple regression.
TTEST	Unpaired t-test using a SAS data set as input.
UNIVCARD	Univariate descriptive statistics using card images as input. Generates, among other statistics, the Kolmogorov-Smirnov test of normality.
UNIVSAS	Univariate descriptive statistics using a SAS data set as input.
UNIVFILE	Univariate descriptive statistics using an "ordinary" operating system file as input. The sample data for this example can be seen in SAS(DATA).

### DATA Step According to the Input Data Source

#### 1. DATA in jobstream

DATA statement;  
 INPUT statement;  
 other SAS programming statements  
 CARDS statement;  
 data lines  
 ;

#### 2. DATA from a System file

DATA statement;  
 INFILE statement;  
 INPUT statement;  
 other SAS programming statements

#### 3. DATA from Existing SAS Data sets

DATA statement;  
 SET, MERGE, or UPDATE statement;  
 other SAS programming statements

Additional programming statements that you can use to build SAS data sets are described in Chapters 3 and 11 of the SAS User's Guide: Basics.

### 4.3.1 DATA Statement

Data must be put into a SAS data set before they can be analyzed using a SAS procedure.

The DATA statement is the first SAS DATA step statement coded, and consists of the keyword DATA followed by a name for the SAS data set created. If the name is omitted, SAS will assign a name to the data set.

DATA name1;

In order to create a new SAS data set, SAS reads observations from the input data, which can have several sources: (1) data lines incorporated into the SAS job using the CARDS and INPUT statements, (2) data lines from a system file made available to the SAS job by the INFILE and INPUT statements, and (3) observations from a previously created SAS data set made available to the SAS job by the SET, MERGE, or UPDATE statements.

One thing to remember when working with SAS data sets is the file structure in which they are stored (refer to Chapter 12, SAS User's Guide: Basics). In the example jobstream on page 41, line 11, the data set which is noted by "DSN=" is referred to as a SAS data library. This data library may contain several individual SAS data sets.

### 4.3.2 INPUT Statement

The INPUT statement reads values into each variable of the input list and describes each value's type and location in the input data lines. There are three input modes; they may be used individually or combined in the same INPUT statement.

(1) List input describes the input as format-free. The variables are simply listed in the order in which they appear on the data lines. For example:

```
INPUT STATION $ DEPTH TEMP;
```

tells SAS that each data line has three values separated from each other by at least one blank space. The first value corresponds to the variable STATION; the second value, to DEPTH; and the third, to TEMP. With list input, missing values must be represented by periods.

In order to read a character value, a dollar sign must appear after the variable name on the INPUT statement (i.e., STATION \$). However, this automatically assigns a length of eight characters to the variable. If this is unacceptable, you can specify a format for the variables according to Chapter 3 of the SAS User's Guide: Basics.

(2) If the data values for each variable occupy the same columns on each data line, column input may be appropriate. To use this mode, first list the variable's name, then the columns that value might occupy. Values may be read in any order, and they can also be reread to give them different variable names. The previous example, restated for column input, might be:

```
INPUT STATION $ 1-8 DEPTH 10-13 TEMP 15-18;
```

This tells SAS that columns 1 through 8 contain the value for the character variable STATION, columns 10 through 13, the value for DEPTH, and columns 15 through 18, the TEMP value.

It is important that the format allow for the largest value in a variable's range, so SAS doesn't inadvertently read only part of a value. However, if the value doesn't fill all the columns you have specified, SAS will ignore blanks both before and after the value.

(3) The third input mode is formatted input which uses the pointer to tell SAS where a data value is located on a data line. The pointer is a mechanism SAS uses to keep track of its position while reading input lines. To use this input mode, the pointer is moved to the column where the value begins, the variable's name is specified, then a format (which describes the value's width) is given. For example:

```
INPUT @10 DEPTH 4.;
```

tells SAS to move the pointer to column 10, then read a depth value that occupies four columns. For a more detailed discussion of formatted input, refer to the SAS User's Guide: Basics.

### 4.3.3 CARDS Statement

If the data values will be entered as lines within your jobstream, or if the data are actually on cards, a CARDS statement is required after the INPUT statement. This statement, whose form is simply

CARDS;

signals SAS that the data cards will be encountered next. The first line after the data value cards must include a semicolon to tell SAS it has reached the end of the data cards.

### 4.3.4 INFILE Statement

If a new SAS data set is to be created using a previously catalogued data set (that is not a SAS data set) as the source of observations, you must tell both the IBM system and SAS where to find the data values. A DD statement in the JCL (see Section 3.3.3) tells the IBM system which data set contains the data values. The SAS INFILE statement tells SAS where the data are located by specifying the DD name of the data set containing the data. A brief form of this statement, which allows its major function of file identification, is

INFILE DDname;

For example, if the data were in data set CN.EPAWOB.WSDG.EXAMPLE, the jobstream should contain statements similar to:

```
//VALUES DD DSN=CN.EPAWOB.WSDG.EXAMPLE,DISP=SHR
DATA;
INFILE VALUES;
INPUT @10 DEPTH 4.;
```

The INFILE statement must appear before the INPUT statement that describes the data lines, and it is very important that the DDname and the name on the INFILE statement are identical.

### 4.3.5 SET Statement

The SET statement reads observations from existing SAS data sets. It may be used to read, subset, concatenate, or interleave observations from existing SAS data sets into a new SAS data set.

The basic form of this statement is:

SET data set name;

The data set name may be omitted, in which case SAS will use the most recently created data set as the source of observations.

If all observations from the old data set are not to be included in the new data set, use a subsetting IF statement. Its form is:

IF expression;

The expression is evaluated for each observation. If it is true, the observation is included in the new data set, if it is false, the observation is deleted and SAS returns to the beginning of the DATA step. For example, to create a data set which contains observations only from the year 1972, the jobstream might include these statements:

```
DATA NAME2;
SET NAME1;
IF YEAR=1972;
```

#### 4.3.6 MERGE Statement

The MERGE statement combines observations from two or more SAS data sets into one new SAS data set. When each data set contains a common variable, use the BY statement to match observations (the data sets must already be sorted by the common variable). In order to join the first observation in one data set with the first observation in another data set, then the second observations, and so on, omit the BY statement.

For example, if two data sets contained different lab data for the same set of stations, all data describing each station could be combined with these statements:

```
PROC SORT DATA=DATA1;
  BY STATION;
PROC SORT DATA=DATA2;
  BY STATION;
DATA DATA3;
  MERGE DATA1 DATA2;
  BY STATION;
```

#### 4.3.7 UPDATE Statement

The UPDATE statement may also be used to create a new SAS data set from two existing ones. Its function is to update a master file by applying a series of transactions. Both the old master file and the transaction file must be sorted by the same identifying variable. The form of this statement is:

```
UPDATE old-master-data-set transaction-data-set;
```

A general example of the use of this statement follows:

```
DATA OLDMSTR;
  INPUT data descriptions;
  CARDS;
  master data records
;PROC SORT; BY ID;

DATA TRANS;
  INPUT data descriptions;
  CARDS;
  transaction records
;PROC SORT; BY ID;
```

```
DATA NEWMSTR;
  UPDATE OLDMSTR TRANS; BY ID;
```

The output data set (NEWMSTR) contains one observation for each observation in the old master file (OLDMSTR), as well as one observation for any transaction observation (TRANS) that was not matched with a master observation.

Additional program statements that you can use to build SAS data sets are described in Chapters 3 and 11 of the SAS User's Guide: Basics.

#### 4.4 The PROC Step

At this point, the tools required to get data into a SAS data set, whether they are on cards, in a system file, or in another SAS data set have been described. SAS procedures can now be used to analyze and process that data set. The statements that tell SAS to analyze a data set comprise the PROC step.

When you issue a PROC statement, SAS automatically performs that procedure on the most recently created data set. To override that default, specify the name of the data set to be processed with the DATA= option in the PROC statement e.g.,

```
PROC UNIVARIATE DATA=FILE3;
```

When the SAS package was installed at the NCC, SAS system options were set to appropriate values for the NCC. The OPTIONS procedure will print a list of the current values for the system options. Statements may be included in the jobstream to alter these values to better suit a particular situation. See OPTIONS statement in Chapter 11 of the SAS User's Guide: Basics.

Each procedure and its options are described in detail in either the SAS User's Guide: Statistics, or SAS User's Guide: Basics.

#### 4.5 SAS Analysis via STORET Keywords

##### 4.5.1 Introduction

STORET User Assistance has developed an interface between STORET and SAS. STORET users may now include SAS in their STORET runs by using the keywords SASPARMS and STOPSAS. Documentation regarding this capability is contained in the data set

```
STORET.HELP.SAS
```

The data set also describes how you can obtain a printed copy of the documentation.

When SAS is used as an "add-on" to any STORET program, it has the ability to read any of the files produced by the STORET retrieval. However, you must code appropriate INFILE and INPUT statements to give SAS access to the STORET data.

One advantage to this procedure, which can be used with any STORET program, is that you are not required to code JCL. However, you should realize that a STORET retrieval using this facility will not save the data which is retrieved. For example, if you need more than one SAS analysis on the same data in different jobs, you will have to duplicate the STORET retrieval for each job.

#### 4.5.2 Interface Example

Box 8 includes a jobstream created by using the STORET %RET command under a TSO session. The %SCAN command was used to check for syntax errors and to submit the jobstream. A line-by-line general explanation of the jobstream, and the output generated follows:

SAS/STORET INTERFACE EXAMPLE

```

00000010      PGM=RET,PURP=104/FED,MORE=SAS,PRT=NO,
00000020      A=1110NET,S=070006,P=10,P=300,BD=60,ED=62,
00000030      SASPARMS=BEGIN,
00000040      OPTIONS S=72 LS=71;
00000050      DATA;
00000060      INFILE FCF;
00000070      INPUT @26 YY $2. @;
00000080      IF YY='99' THEN DELETE; DROP YY;
00000090      FORMAT DATE YYMMDD8.;
00000100      INPUT @1 AGN $8.
00000110      @9 STN $15.
00000120      @26 DATE YYMMDD6.
00000130      @36 TEMP C RB4.
00000140      DISS OX RB4.;
00000150      PROC SORT; BY AGN;
00000160      PROC PLOT; BY AGN;
00000170      PLOT TEMP C * DATE/VPOS=40;
00000180      PLOT DISS OX * DATE/VPOS=40;
00000190      TITLE 'FIGURE 3';
00000200      STOPSAS;
00000201      ./EPAWOB JOB (WSDGSTORPUUU,MWOB),STORET,NOTIFY=EPAWOB,
00000202      ./ MSGLEVEL=(1,1),PRTY=2
00000203      **ROUTE PRINT HOLD
00000204      **JOBPARM LINES=10

```

<u>LINE</u>	<u>EXPLANATION</u>
10-20	Typical STORET retrieval for one station, two parameters, for three years. MORE=SAS requests an FCF. PRT=NO requests that the data values not be listed.
30	SAS/STORET boundary.
40	Resets SAS system options: S=72        input restricted to 72 columns LS=71       output restricted to 71 columns
50	Beginning of DATA step.
60	Use Further Computational File as input data.
70-80	Read year; if it is equal to 99 (STORET delimiter record), delete the record.
90	Define default format for outputting date.
100-140	Input and format specifications.

<u>LINE</u>	<u>EXPLANATION</u>
150	Beginning of PROC step.
160-180	Requests two plots and specifies number of lines on the vertical axis.
190	Defines title for each plot; replaces default title, S T A T I S T I C A L   A N A L Y S I S   S Y S T E M .
200	STORET keyword that signifies end of SAS job
201-204	JCL generated by STORET to complete the job

## 5.0 References

IBM Corporation. 1979. OS/VS Message Library: VS2 System Messages. Release 3.8. Eighth Edition. Poughkeepsie, N.Y.

Ingwersen, James B. January 1981. Statistical Analysis Using SAS at the USEPA National Computer Center. WSDG Report WSDG-AD-00001. USDA Forest Service, Watershed Systems Development Group, 3825 East Mulberry Street, Fort Collins, Colo. 88 p.

International Business Machines. 1979. OS/VS2 MVS JCL. Poughkeepsie, N.Y.

National Computer Center. 1982. NCC-IBM User's Guide. Research Triangle Park, N.C.

On-Line Business Systems, Inc. 1979. Introduction to OBS WYLBUR. Second Edition. San Francisco, Calif.

Ponce, Stanley L. 1980. Statistical Methods Commonly Used in Water Quality Data Analysis. WSDG Report WSDG-TP-00001. USDA Forest Service, Watershed Systems Development Group, 3825 East Mulberry Street, Fort Collins, Colo. 147 p.

SAS Institute Inc. 1978. SAS Introductory Guide. Cary, N.C.

SAS Institute. 1982. SAS User's Guide: Basics. Cary, N.C.

SAS Institute. 1982. SAS User's Guide: Statistics. Cary, N.C.

U.S. Environmental Protection Agency. STORET User Handbook. Office of Water and Hazardous Materials, Washington D.C.



## Appendix A

STORET Retrieval and SAS Analysis

Using WYLBUR



Problem: Retrieve data values for two parameters (temperature and dissolved oxygen) from one station above Reudi Dam and one station below Reudi Dam. Store the data in a SAS data set and catalog the data set. Perform the CONTENTS procedure to document the SAS data set then the UNIVARIATE procedure to test the data for normality.

User response is underlined

Jobstream:

PLEASE TYPE YOUR TERMINAL IDENTIFIER

-2401-161-

PLEASE LOG IN:IBMEPA1;NCC

P 16

IBM1 IS ON LINE

OBS

OBS WYLBUR 6.0 AT EPA TERMINAL TTY060 17:16:40 09/21/82

USER ID? EPAWOB

PASSWORD (CURR(/NEW))? PASS

ACCTUID? WSDGSTORP

BIN/BOX? MWOB

* 09/14/82: NEW SYNC SORT LEVEL IMPLEMENTATION	-SEE NEWS ALERT2 *
* 09/16/82: WYLBUR RELEASE 6.A IMPLEMENTATION	-SEE NEWS ALERT3 *
* 09/20/82: UPDATE AVAIL. TO NCC-IBM USER'S GUIDE	-SEE NEWS ALERT11*

CN.EPAWOB.WSDG.QWIK. NOT ON CATLG 0 LEVELS OK

CN.EPAWOB.WSDG.ACTIVE NOT ON CATLG 0 LEVELS OK

CN.EPAWOB.WSDG.LIB NOT ON CATLG

EXEC BREAK

COMMAND? USE \$CN.EPAWOA.WSDG.SAS(DATASET)

COMMAND? LIST

```

1.  //EPAXXX JOB (AAAAAUID,MXXX),NAME,PASSWORD=WWW,TIME=T,PRTY=N
2.  /*
3.  /* THIS RUNSTREAM WILL PERFORM A STORET RETRIEVAL, CATALOG
4.  /* A SAS DATA SET CONTAINING THE STORET DATA, AND DO A
5.  /* CONTENTS PROCEDURE ON THE SAS DATA SET.
6.  /*
7.  /*ROUTE PRINT RMTNNN
8.  //STEPSTOR EXEC WQDIST
9.  //DIST.CARDFD DD *
10. PGM=RET,PURP=303/AGENCY,MORE=SAS,
11. A=GGGGGGGG,S=SSSSSSSSSSSSSS,S=SSSSSSSSSSSSSS,
12. P=PPPPP,P=PPPPP,P=PPPPP,
13. /* THE ABOVE STATEMENTS PERFORM THE STORET RETRIEVAL.
14. /* THE FOLLOWING STATEMENTS CREATE AND CATALOG A SAS
15. /* DATA SET CONTAINING THE RETRIEVED STORET DATA.
16. //STEPSAS EXEC SAS
17. //STORET DD DSN=&FCF,DISP=(OLD,PASS)
18. //SAVESAS DD UNIT=DISK,DISP=(NEW,CATLG,DELETE),
19. //    SPACE=(19069,(5,5),RLSE),DSN=CN.EPAXXX.AAAA.FILENAME
20. //SYSIN DD *
21. DATA SAVESAS.SASNAME;
22. INFILE STORET LENGTH=L;
23. INPUT @;
24. IF L LT 305 THEN DELETE;
25. INPUT @9 STA $15. DATE 26-31 YY 26-27 MM 28-29 DD 30-31
26. TIME 32-35 @36 PARM1 RB4. PARM2 RB4. PARM3 RB4.;
27. * YOU MAY HAVE UP TO 50 PARAMETERS IN THE ABOVE
28. STATEMENT. BE SURE THEY APPEAR IN THE SAME ORDER
29. AS THEY WERE IN THE STORET RETRIEVAL (LINE 13).;
30. IF YY=99 THEN DELETE;
31. IF PARM1 LT 1.E-10 AND PARM1 GT 0 THEN PARM1=.;
32. IF PARM2 LT 1.E-10 AND PARM2 GT 0 THEN PARM2=.;
33. IF PARM3 LT 1.E-10 AND PARM3 GT 0 THEN PARM3=.;
34. * INCLUDE A LINE LIKE THE ABOVE FOR EACH PARAMETER
35. RETRIEVED. ;
36. PROC CONTENTS;

```

37. TITLE CONTENTS OF FILENAME.SASNAME;  
 38. \*  
 39. THE FOLLOWING CHANGES NEED TO BE MADE PRIOR TO  
 40. RUNNING THIS JOB:  
 41. -  
 42. CHANGE CHARACTERS IN LINE NO. TO  
 43.  
 44. XXX 1,19 CORRECT USER ID  
 45. AAAAAUID 1 ACCOUNT UTILIZATION ID  
 46. NAME 1 PROGRAMMER'S NAME  
 47. (MAXIMUM 20 CHARACTERS)  
 48. WWW 1 YOUR CURRENT PASSWORD  
 49. =T 1 UP TO 4 NUMERIC DIGITS  
 50. (DEFAULT 30 SECONDS)  
 51. =N 1 RANGE FROM 1 TO 5  
 52. (DEFAULT IS 2)  
 53. NNN 7 HIGH-SPEED PRINTER SITE ID  
 54. AGENCY 10 YOUR AGENCY NAME  
 55. (ALPHA CHARACTERS)  
 56. GGGGGGGG 11 YOUR AGENCY STORET CODE  
 57. (1 TO 8 NUMERIC CHARACTERS)  
 58. SSSSSSSSSSSSSS 11 STATION CODES (1-15 CHARAC.)  
 59. (ADD OR DELETE CODES AS  
 60. NECESSARY FOR RETRIEVAL)  
 61. PPPPP 12 PARAMETER CODES (1-5 CHARAC.)  
 62. (UP TO 50 PARAMETERS AS DESIRED)  
 63. AAAA 19 YOUR ACCOUNT CHARACTERS  
 64. PARM1-PARM3.. 26, 31-33 PARAMETER NAMES  
 65. FILENAME 19, 37 ANY NAME OF YOUR CHOICE.  
 66. THIS WILL BE THE IBM DATA  
 67. FILE WHICH HOLDS YOUR SAS  
 68. DATA SET.  
 69. SASNAME 21, 37 ANY NAME OF YOUR CHOICE.  
 70. THIS WILL BE THE NAME OF  
 71. YOUR SAS DATA SET WHICH WILL  
 72. RESIDE IN THE IBM DATA FILE  
 73. 'FILENAME'.  
 74.

75. IF YOU NEED ADDITIONAL ASSISTANCE, PLEASE CONTACT THE WSDG AT

76. (303) 482-0356 OR FTS 323-1417.;

COMMAND? DELETE 2/6

COMMAND? DELETE 13/15

COMMAND? DELETE 27/29

COMMAND? DELETE 34/35

COMMAND? DELETE 38/76

COMMAND? NUMBER

24. - LAST LINE.

COMMAND? LIST

1. //EPAXXX JOB (AAAAAUID,MXXX),NAME,PASSWORD=WWW,TIME=T,PRTY=N
2. /\*ROUTE PRINT RMTNNN
3. /STEPSTOR EXEC WQDIST
4. //DIST.CARDFD DD \*
5. PGM=RET,PURP=303/AGENCY,MORE=SAS,
6. A=GGGGGGGGG,S=SSSSSSSSSSSSSSSS,S=SSSSSSSSSSSSSSSS,
7. P=PPPPP,P=PPPPP,P=PPPPP,
8. //STEPSAS EXEC SAS
9. //STORET DD DSN=&FCF,DISP=(OLD,PASS)
10. //SAVESAS DD UNIT=DISK,DISP=(NEW,CATLG,DELETE),
11. // SPACE=(19069,(5,5),RLSE),DSN=CN.EPAXXX.AAAA.FILENAME
12. //SYSIN DD \*
13. DATA SAVESAS.SASNAME;
14. INFILE STORET LENGTH=L;
15. INPUT @;
16. IF L LT 305 THEN DELETE;
17. INPUT @9 STA \$15. DATE 26-31 YY 26-27 MM 28-29 DD 30-31
18. TIME 32-35 @36 PARM1 RB4. PARM2 RB4. PARM3 RB4.;
19. IF YY=99 THEN DELETE;
20. IF PARM1 LT 1.E-10 AND PARM1 GT 0 THEN PARM1=.;
21. IF PARM2 LT 1.E-10 AND PARM2 GT 0 THEN PARM2=.;
22. IF PARM3 LT 1.E-10 AND PARM3 GT 0 THEN PARM3=.;
23. PROC CONTENTS;
24. TITLE CONTENTS OF FILENAME.SASNAME;

COMMAND? CHANGE 'XXX' TO 'WOB' IN ALL

1. //EPAWOB JOB (AAAAAUID,MWOB),NAME,PASSWORD=WWW,TIME=T,PRTY=N

11. // SPACE=(19069,(5,5),RLSE),DSN=CN.EPAWOB.AAAA.FILENAME

COMMAND? CHANGE 'AAAAAUID' TO 'WSDGSTORP' IN 1

1. //EPAWOB JOB (WSDGSTORP,MWOB),NAME,PASSWORD=WWW,TIME=T,PRTY=N

COMMAND? CHANGE 'NAME' TO 'EICHIN' IN 1

1. //EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,PASSWORD=WWW,TIME=T,PRTY=N

COMMAND? CHANGE 'WWW,TIME=T' TO 'PASS' IN 1

1. //EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,PASSWORD=PASS,PRTY=N

COMMAND? CHANGE '=N' TO '=3' IN 1

1. //EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,PASSWORD=PASS,PRTY=3

COMMAND? CHANGE 'NNN' TO '184' IN 2

2. /\*ROUTE PRINT RMT184

COMMAND? CHANGE '303/AGENCY' TO '104/FED' IN 5

5. PGM=RET,PURP=104/FED,MORE=SAS

COMMAND? REPLACE 6

6. ? A=113FORS2,S=41-001,S=43-001,

COMMAND? REPLACE 7

7. ? P=10,P=300,

COMMAND? CHANGE 'AAAA.FILENAME' TO 'WSDG.RET1' IN 11

11. // SPACE=(19069,(5,5),RLSE),DNS=CN.EPAWOB.WSDG.RET1

COMMAND? CHANGE 'PARM1' TO 'TEMP' IN 18

18. TIME 32-35 @36 TEMP RB4. PARM2 RB4. PARM3 RB4.;

COMMAND? CHANGE 'PARM2 RB4. PARM3 RB4.' TO 'DISSOX RB4.' IN 18

18. TIME 32-35 @36 TEMP RB4. DISSOX RB4.;

COMMAND? CHANGE 'PARM1' TO 'TEMP' IN 20

20. IF TEMP LT 1.E-10 AND TEMP GT 0 THEN TEMP=.;

COMMAND? CHANGE 'PARM2' TO 'DISSOX' IN 21

21. IF DISSOX LT 1.E-10 AND DISSOX GT 0 THEN DISSOX=.;

COMMAND? DELETE 22

COMMAND? CHANGE 'FILENAME' TO 'RET1' IN 24

24. TITLE CONTENTS OF RET1.SASNAME;

COMMAND? INSERT 25 26 27

25. ? PROC UNIVARIATE PLOT NORMAL;

26. ? TITLE UNIVARIATE PROCEDURE ON RET1;

27. ? VAR TEMP DISSOX;

COMMAND? NUMBER

26. - LAST LINE.

COMMAND? LIST

1. //EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,PASSWORD=PASS,PRTY=3
2. /\*ROUTE PRINT RMT184
3. //STEPSTOR EXEC WQDIST
4. //DIST.CARDFD DD \*
5. PGM=RET,PURP=104/FED,MORE=SAS,
6. A=113FORS2,S=41-001,S=43-001,
7. P=10,P=300,
8. //STEPSAS EXEC SAS
9. //STORET DD DSN=&FCF,DISP=(OLD,PASS)
10. //SAVESAS DD UNIT=DISK,DISP=(NEW,CATLG,DELETE),
11. // SPACE=(19069,(5,5),RLSE),DSN=CN.EPAWOB.WSDG.RET1
12. //SYSIN DD \*
13. DATA SAVESAS.SASNAME;
14. INFILE STORET LENGTH=L;
15. INPUT @;
16. IF L LT 305 THEN DELETE;
17. INPUT @9 STA \$15. DATE 26-31 YY 26-27 MM 28-29 DD 30-31
18. TIME 32-35 @36 TEMP RB4. DISSOX RB4.;
19. IF YY=99 THEN DELETE;
20. IF TEMP LT 1.E-10 AND TEMP GT 0 THEN TEMP=.;
21. IF DISSOX LT 1.E-10 AND DISSOX GT 0 THEN DISSOX=.;
22. PROC CONTENTS;
23. TITLE CONTENTS OF RET1.SASNAME;
24. PROC UNIVARIATE PLOT NORMAL;
25. TITLE UNIVARIATE PROCEDURE ON RET1;
26. VAR TEMP DISSOX;

COMMAND? SAVE NEWRET ON USER40

CN.EPAWOB.WSDG.NEWRET SAVED AND CATLG'D ON USER40

COMMAND? RUN FETCH

JOB 4592 EPAWOB SUBMITTED

COMMAND? LOCATE

JOB 4592 EPAWOB92 AW FETCH

COMMAND? FETCH 4592 CLR

COMMAND? LIST 'ERROR'

VOID RANGE.

COMMAND? PRINT 4592 DEST RM184

COMMAND? LOC

JOB 4592 EPAWOB92 ON OUT RM184

COMMAND? LOGOFF CLR

EPAWOB/WSDG OFF OBS WYLBUR AT 17:37:26 09/21/82 (82.264)

-USAGE- CONN MNS: 1.46 CPU SECS .64 DA I/O: 8 TERM I/O: 38

\*COSTS\* CONN: \$.14 CPU: \$.30 EXCP: \$.14 \*TOTAL\*: \$.58

END OF SESSION

Example Output

STORET RETRIEVAL

PROC CONTENTS

PROC UNIVARIATE



STORET RETRIEVAL DATE 83/03/16

41-001  
39 21 45.0 106 43 58.0 1  
FRYINGPAN RIVER AT MEREDITH  
08037 COLORADO EAGLE  
COLORADO RIVER BASIN 110391  
UPPER COLORADO RIVER BASIN  
113FORS2 14010004000  
0000 FEET DEPTH CLASS 00 CSN-RSP 0224903-0339485

/TYPA/AMBNT/STREAM

DATE FROM TO	TIME OF DAY	DEPTH FEET	00010 WATER TEMP CENT	00300 DO MG/L
73/05/11	13 30			8.2
75/02/12	10 30		0.0	
76/06/07	11 00		7.0	
76/07/13	13 50		16.0	9.0
76/07/26	10 15		12.0	
76/08/11	13 35		13.0	9.0
76/08/16	13 35		15.0	10.0
76/09/14	12 35		13.0	
76/09/24	09 50		8.0	
77/10/26	15 00			10.0
78/01/25	12 00		0.5	11.0

STORET RETRIEVAL DATE 83/03/16

43-001  
39 21 56.0 106 49 30.0 2  
FRYINGPAN R NEAR REUDI  
08037 COLORADO  
COLORADO RIVER BASIN 110391  
UPPER COLORADO RIVER BASIN  
113FORS2  
0000 FEET DEPTH CLASS 00 CSN-RSP 0224907-0339492

/TYPA/AMBNT/STREAM

DATE FROM TO	TIME OF DAY	DEPTH FEET	00010 WATER TEMP CENT	00300 DO MG/L
73/05/11	11 00			12.6
75/02/12	09 45		3.0	
76/06/07	10 30		5.0	
76/07/13	13 10		7.0	11.0
76/07/26	09 35		6.0	
76/08/11	14 35		6.0	12.0
76/08/16	14 30		7.0	13.0
76/09/14	13 30		7.0	
76/09/24	09 20		7.0	

CONTENTS OF SAS DATA SET SAVESAS.SASNAME

TRACKS USED=2 SUBEXTENTS=1 OBSERVATIONS=20 CREATED BY OS JOB EPAWOB49  
ON CPUID 01-0168-060353 AT 16:41 WEDNESDAY, MARCH 16, 1983  
BY SAS RELEASE 82.2B DSN=CN.EPAWOB.WSDG.RET1  
INFILE(DSN=SYS83075.T162817.RA000.EPAWOB49.FCF VOL=SER=WORK55)  
BLKSIZE=19054 LRECL=75 OBSERVATIONS PER TRACK=254 GENERATED BY DATA

ALPHABETIC LIST OF VARIABLES

#	VARIABLE	TYPE	LENGTH	POSITION	FORMAT	INFORMAT
2	DATE	NUM	8	19		
5	DD	NUM	8	43		
8	DISSOX	NUM	8	67		
4	MM	NUM	8	35		
1	STA	CHAR	15	4		
7	TEMP	NUM	8	59		
6	TIME	NUM	8	51		
3	YY	NUM	8	27		

```

+----- SOURCE STATEMENTS -----
[DATA SAVESAS.SASNAME;
[OPTIONS PAGESIZE=64 LINESIZE=75;
[INFILE STORET LENGTH=L;
[INPUT @;
[IF L<305 THEN DELETE;
[INPUT @9 STA $15. DATE 26-31 YY 26-27 MM 28-29 DD 30-31
[TIME 32-35 @36 TEMP RB4. DISSOX RB4.;
[IF YY=99 THEN DELETE;
[IF TEMP<1.E-10 AND TEMP>0 THEN TEMP=.;
[IF DISSOX<1.E-10 AND DISSOX>0 THEN DISSOX=.;
+-----

```

16:41 WEDNESDAY, MARCH 16, 1983

## UNIVARIATE

VARIABLE=TEMP

## MOMENTS

N	17	SUM WGTS	17
MEAN	7.79412	SUM	132.5
STD DEV	4.64711	VARIANCE	21.5956
SKEWNESS	0.194585	KURTOSIS	-0.504351
USS	1378.25	CSS	345.529
CV	59.6232	STD MEAN	1.12709
T:MEAN=0	6.91527	PROB>[T[	0.0001
SGN RANK	68	PROB>[S[	.000459626
NUM ]= 0	16		
W:NORMAL	0.933658	PROB<W	0.315

## QUANTILES(DEF=4)

100% MAX	16	99%	16
75% Q3	12.5	95%	16
50% MED	7	90%	15.2
25% Q1	5.5	10%	0.4
0% MIN	0	5%	0
		1%	0
RANGE	16		
Q3-Q1	7		
MODE	7		

## EXTREMES

LOWEST	HIGHEST
0	12
0.5	13
3	13
5	15
6	16

MISSING VALUE	.
COUNT	3
% COUNT/NOBS	15.00

16:41 WEDNESDAY, MARCH 16, 1983

## UNIVARIATE

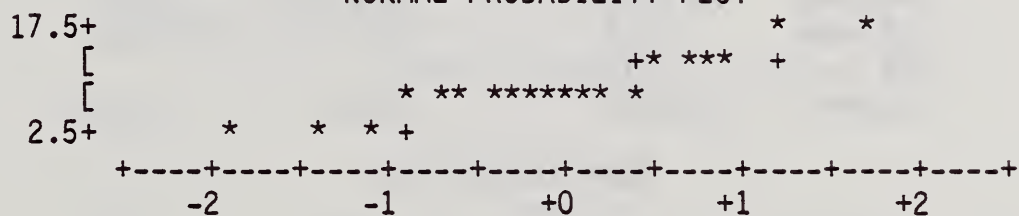
VARIABLE=TEMP

STEM	LEAF	#	BOXPLOT
1	56	2	[
1	233	3	+-----+
0	566777778	9	*--+-*
0	003	3	[

-----+-----+-----+-----+

MULTIPLY STEM.LEAF BY 10\*\*+01

## NORMAL PROBABILITY PLOT



16:41 WEDNESDAY, MARCH 16, 1983

## UNIVARIATE

VARIABLE=DISSOX

## MOMENTS

N	10	SUM WGTS	10
MEAN	10.58	SUM	105.8
STD DEV	1.62056	VARIANCE	2.62622
SKEWNESS	0.119644	KURTOSIS	-1.17954
USS	1143	CSS	23.636
CV	15.3172	STD MEAN	0.512467
T:MEAN=0	20.6452	PROB>[T[	0.0001
SGN RANK	27.5	PROB>[S[	0.00582502
NUM ]= 0	10		
W:NORMAL	0.952204	PROB<W	0.667

## QUANTILES(DEF=4)

100% MAX	13	99%	13
75% Q3	12.15	95%	13
50% MED	10.5	90%	12.96
25% Q1	9	10%	8.28
0% MIN	8.2	5%	8.2
		1%	8.2
RANGE	4.8		
Q3-Q1	3.15		
MODE	9		

## EXTREMES

LOWEST	HIGHEST
8.2	11
9	11
9	12
10	12.6
10	13

MISSING VALUE	.
COUNT	10
% COUNT/NOBS	50.00

16:41 WEDNESDAY, MARCH 16, 1983

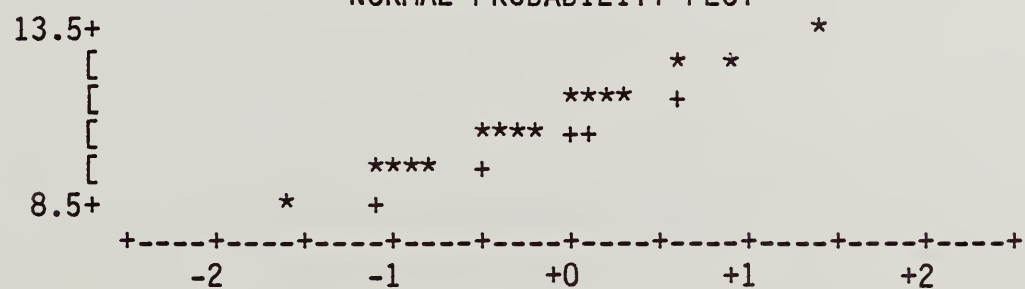
## UNIVARIATE

VARIABLE=DISSOX

STEM	LEAF	#	BOXPLOT
13	0	1	[
12	06	2	+-----+
11	00	2	[ [
10	00	2	*--+-*
9	00	2	+-----+
8	2	1	[

-----+-----+-----+

## NORMAL PROBABILITY PLOT





## Appendix B

### SAS Analysis of Previously Created SAS Data Set Using TSO



Problem: Using the SAS data set created and cataloged by the procedure in Appendix A as input, perform the CORR procedure, and do analysis by station. User response is underlined.

Jobstream:

PLEASE TYPE YOUR TERMINAL IDENTIFIER

-2401-001-

PLEASE LOG IN:IBMEPA1;NCC

P 8

IBM1 IS ON LINE

TSO

ENTER LOGON

LOGON EPAWOB/PASS ACCT(WSDGSTORP)

ICH7000II EPAWOB - LAST ACCESS AT 16:57:16 ON WEDNESDAY, SEPTEMBER 22, 1982

EPAWOB LOGON IN PROGRESS 17:18:16 ON SEPTEMBER 22, 1982

NO BROADCAST MESSAGES

WELCOME TO TSO

YOU ARE NOW IN THE STORET ENVIRONMENT

READY

EDIT REUD NEW

INPUT

```
00010    //EPAWOB JOB (WSDGSTORP,MWOB),EICHIN,PASSWORD=PASS,PRTY=3
00020    /*ROUTE PRINT RMT255
00030    //      EXEC SAS
00040    //REUDI DD DSN=CN.EPAWOB.WSDG.RET1,DISP=OLD
00050    //SYSIN DD *
00060    DATA;
00070    SET REUDI.SASNAME;
00080    OPTIONS PAGESIZE=64 LINESIZE=75;
00090    PROC SORT; BY STA;
00100    PROC CORR; BY STA;
00110    VAR TEMP DISOX;
00120    TITLE1 CORRELATION ANALYSIS OF TEMPERATURE AND DISSOLVED OXYGEN;
```

00130 TITLE2 ABOVE AND BELOW REUDI DAM - COLORADO;

00140

QED

END SAVE

SAVED

READY

SUBMIT REUD

JOB EPAWOB(JOB09898) SUBMITTED

READY

18.12.48 JOB 9898 \$HASPI65 EPAWOB98 ENDED AT NCCIBM CN(00)

QUEUE

Q0800 WELCOME TO QUEUE - ENTER: "H" FOR COMMAND LIST

QUEUE

FETCH 9898

1 JES2 JOB LOG -- SYSTEM EPA1 --

NODE NCCIBM

QUEUE

FA 'ERROR'

QUEUE

ROUTE 9898 RM184

Q3009 JOB SCHEDULED FOR REMOTE

QUEUE

END

READY

LOGOFF

Example Output

PROC CORR



CORRELATION ANALYSIS OF TEMPERATURE AND DISSOLVED OXYGEN  
 ABOVE AND BELOW REUDI DAM - COLORADO

1

16:24 WEDNESDAY, MARCH 16, 1983

STA=41-001

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
TEMP	9	9.3888889	5.9465209	84.500000	0	16.000000
DISSOX	6	9.5333333	0.9933110	57.200000	8.200000	11.000000

CORRELATION COEFFICIENTS / PROB > [R] UNDER HO:RHO=0  
 / NUMBER OF OBSERVATIONS

	TEMP	DISSOX
TEMP	1.00000	-0.84107
	0.0000	0.1589
	9	4
DISSOX	-0.84107	1.00000
	0.1589	0.0000
	4	6

CORRELATION ANALYSIS OF TEMPERATURE AND DISSOLVED OXYGEN  
 ABOVE AND BELOW REUDI DAM - COLORADO

2

16:24 WEDNESDAY, MARCH 16, 1983

STA=43-001

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
TEMP	8	6.000000	1.4142136	48.000000	3.000000	7.000000
DISSOX	4	12.150000	0.8698658	48.599999	11.000000	13.000000

CORRELATION COEFFICIENTS / PROB > [R[ UNDER HO:RHO=0  
 / NUMBER OF OBSERVATIONS

	TEMP	DISSOX
TEMP	1.00000 0.0000 8	0.00000 1.0000 3
DISSOX	0.00000 1.0000 3	1.00000 0.0000 4

2

